Crowdfunding Platform

Create a full stack crowdfunding platform using HTML, CSS, JavaScript, JSON Server, and JSON Server Auth to simulate REST API functionality to enable Users to launch campaigns, or to pledge support, and Admins to moderate content. Core features include role-based access, campaign management, real-time data interactions, and mock payment flows.

Deadline (05-09-2025)

Requirements

1. Admin (40%)

Backend Interaction: Manage data using the admin dashboard.
Key Features:
- **User Management**:
  - Ban users by setting `isActive: false` in `users[]`.
- **Campaign Moderation**:
  - Approve/reject campaigns (update `isApproved: true` in `campaigns[]`).
  - Delete campaigns violating guidelines.
- **Data Access**:
  - View all users, campaigns, and pledges in `db.json`.

---

2. Registered User (55%)

- **Create campaign:**

Frontend Interaction: Use JavaScript `fetch()` to communicate with JSON Server.
Key Features:
- **Campaign Creation**:
  - POST new campaigns to `/campaigns` (fields: `title`, `description`, `goal`, `deadline`, `rewards[]`).
  - Upload images via Base64 encoding (stored as strings in `campaigns[]`).
- **Campaign Updates**:
  - PATCH campaign details (e.g., `deadline`, `rewards`) using `campaigns/:id`.
  - GET pledge data from `/pledges?campaignId=:id`.

- **Support campaign:**

Frontend Interaction: Browse campaigns and pledge via forms.
Key Features:
- **Account Management**:
  - Register/login via POST to `/users`.
  - View pledge history (`GET /pledges?userId=:id`).
- **Campaign Interaction**:
  - Filter campaigns using `GET /campaigns?category=:category&_sort=deadline`.
  - Submit pledges via POST to `/pledges` (fields: `amount`, `rewardId`, `campaignId`, `userId`).

---

4. Anonymous User (5%)

Access: Read-only mode (no authentication required).
Features:
- Browse campaigns: `GET /campaigns?isApproved=true`.
- Search campaigns using URL query params (e.g., `GET /campaigns?q=music`).

Technology Stack

Frontend: HTML, CSS or Sass (Bonus for proper usage of SASS), and JavaScript (ES6+) only.

Backend
- **JSON Server**:
  - Simulate a REST API with `db.json`.
  - Routes: `/users`, `/campaigns`, `/pledges`.

- **JSON Auth Server**:
  - Create a mock API that supports user authentication to manage user sign-up and login functionalities using a local JSON file to store user data.

Database Schema (db.json)
```json
{
```

```
  "users": [
    {
      "id": 1,
      "name": "Jane Doe",
      "role": "user",
      "isActive": true,
      "email": "jane@example.com",
      "password": "hashed_password"
    }
  ],
  "campaigns": [
    {
      "id": 1,
      "title": "Smart Watch",
      "creatorId": 1,
      "goal": 5000,
      "deadline": "2024-12-31",
      "isApproved": false,
      "rewards": [
        { "id": 1, "title": "Early Bird", "amount": 50 }
      ]
    }
  ],
  "pledges": [
    {
      "id": 1,
      "campaignId": 1,
      "userId": 2,
      "amount": 100,
      "rewardId": 1
    }
  ]
}
```

Additional Requirements:

- **Payments**: Mock payment flow with a confirmation dialog (no real money processing).

- **Version Control:** Use Git and GitHub.

Inspiration websites:

- https://www.gofundme.com
- https://www.kickstarter.com
- https://www.crowdfunding.com